

REMARKS

Claims 1-7 and 21-28 were pending. Claims 1 and 21 have been amended to further clarify the nature of the claimed invention. Support for these amendments may be found in the Specification on at least page 10, lines 8-24 and page 13, line 24 to page 14, line 29. Accordingly, claims 1-7 and 21-28 remain pending subsequent entry of the present amendment.

In the present Office Action, claims 1-7 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Bracha et al. (U.S. 6,687,760, hereinafter "Bracha"), in view of Harrison et al. (U.S. Patent No. 6,128,717, hereinafter "Harrison"). As noted above, Applicant has amended claim 1 to further clarify the nature of the claimed invention. Applicant traverses at least some of the rejections above. Accordingly, Applicant requests reconsideration in view of the following comments.

In Applicant's previous response dated December 12, 2005, Applicant noted the distinct natures of the claimed invention vis-à-vis the cited art. For example, Applicant noted the claimed invention is generally directed to storage systems and entities which may be stored therein. In contrast, Bracha is directed to object oriented programming. While Applicant noted numerous distinctions, the examiner suggests on page 2 of the present Office Action that Bracha's objects and the claimed storage objects "are the same object in that they are both identified the location of the storages." Without conceding this point, Applicant submits that there are further differences between the features of claim 1, as amended, and the cited art.

Claim 1 now reads as follows:

"A method for resolving a storage object's absolute location within a first storage environment to grant access to the storage object, comprising:
receiving a request to access a storage object residing on a non-volatile storage device of the first storage environment;
receiving a storage object reference which corresponds to the storage object;

determining an initial storage management stack level associated with the storage object reference;
in response to determining the storage object reference is not an absolute reference, iterating through one or more additional storage management stack levels beginning with the initial stack level, **wherein each storage management stack level identifies one or more relative extents of storage objects according to a different level of abstraction;** and
translating the storage object reference through each iteration into one or more level-specific relative extents at each storage management stack level until one or more absolute extents are obtained, wherein the one or more absolute extents comprise the storage object's absolute location within the first storage environment.”

As seen from the amendment to claim 1, it is specified that each storage management stack level identifies one or more relative extents of storage objects according to a different level of abstraction. In contrast, Bracha merely discloses a lookup method for determining which implementation of a method of a class to invoke where the class is part of a class hierarchy. More specifically, Bracha discloses

“As shown in FIG. 6a, the lookup method of the present invention begins by determining (602) whether the resolved method (method m of class A 402) has been marked as public/protected. If it has, then it can be accessed and overridden by all subsequent subclasses. If that is the case, then the standard method lookup procedure discussed above may be used to perform (604) the method dispatch. Namely, the class 408 of the target object T 410 is searched for an implementation of the method m. If an implementation is found, then it is invoked. Otherwise, the next superclass 406 is searched for an implementation of method m. If an implementation is found in the superclass 406, then that implementation is invoked. Otherwise, the next superclass 404 is searched. This process continues until either an implementation for method m is found and invoked in one of the subclasses 404, 406, 408, or the resolved class 402 is reached, in which case the resolved method implementation m(1) is invoked.” (Bracha, col. 5, lines 39-56).

As may be seen from the above, Bracha teaches searching successive levels of a class hierarchy for an implementation of a particular method m that is associated with a target object T. However, Bracha fails to mention anything about levels of abstraction or extents of target object T, and especially not level-specific extents.

In the present Office Action (page 3), it is suggested that Harrison discloses “the relative extents in storage (see col. 6, lines 1-8; Fig. 6 and corresponding text, Harrison).” It is further suggested “It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Bracha as taught by Harrison to include the claimed features. The motivation of doing so would have been to improve the techniques for accessibility to objects (col. 2, lines 48-53, Bracha).” While Harrison discloses a storage application programming interface (SAPI) implementation engine that at least in part bases a storage strategy on record extents, Applicant respectfully disagrees that the combination of Bracha and Harrison would produce the claimed features. Applicant also disagrees that a motivation to modify Bracha as taught by Harrison may be found in Bracha.

Harrison is generally directed to an API for data storage and retrieval that takes into account the type and size of data objects. More specifically, Harrison discloses

“Essentially, the SAPI engine 16 learns, discovers or is informed that a particular user data object is of a particular type or category which is meaningful to the drive 10. With this knowledge, the SAPI engine 16 establishes an intelligent mapping paradigm between a conventional primary or device driver LBA linear address space 18 and a new LOA space 20 located within the drive 10 at an indirection level illustrated as the SAPI engine level in FIG. 4. From a global perspective, the mapping from the primary LBA space 18 to the LOA space is a dynamic (one-to-one, one-to-many, many-to-one and many-to-many) mapping paradigm we herein define as SAPI.” (Harrison, col. 8, lines 35-47).

As may be seen from the above, Harrison discloses an SAPI engine that maps LBA space addresses to LOA space at one particular level, the “SAPI engine level in FIG. 4”. Harrison does not disclose that the SAPI identifies record extents at different levels of abstraction. Accordingly, there is no teaching or suggestion that “each storage management stack level identifies one or more relative extents of storage objects according to a different level of abstraction,” as is recited in claim 1. Nor is there any teaching or suggestion of “translating the storage object reference through each iteration

into one or more level-specific relative extents at each storage management stack level,” as is further recited in claim 1.

Also, Applicant submits that the portion of Bracha cited as providing a motivation to combine the references fails to do so. The cited portion of Bracha reads as follows:

“Because this is a somewhat non-intuitive result, and because transitive method override is a desirable property, there is a need for an improved dynamic method dispatch mechanism which is both capable of **enforcing the accessibility constraints** of modular constructs and providing transitive method override.” (Bracha, col. 2, lines 48-53, emphasis added).

Rather than suggesting that there is a need to improve the techniques for accessibility to objects, the cited portion of Bracha suggests that there is a need for an improved dynamic method dispatch mechanism that provides transitive method override as well as enforcing the accessibility constraints of modular constructs. Enforcing accessibility constraints is not the same as improving techniques for accessibility. Further, what is being constrained by Bracha is access to particular *methods* that are part of a class definition, i.e. a description of the type of an object, not access to the objects themselves. The Bracha and Harrison references are quite distinct.

Accordingly, Applicant submits not all of the features of claim 1 are disclosed or suggested by the cited art, whether taken singly or in combination with one another, and a prima facie case of obviousness is not established. Therefore, claim 1 is patentably distinct from the combined cited art. As each of dependent claims 2-7 includes at least the features of independent claim 1, each of these claims is believed patentable for at least the above reasons.

In addition to the above, claims 21-28 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Bracha et al. (U.S. 6,687,760, hereinafter “Bracha”), in view of Jackowski et al. (U.S. Patent No. 6,141,686, hereinafter “Jackowski”). Applicant submits that claim 21, as amended, recites features neither taught nor suggested by the cited art. Accordingly, Applicant requests reconsideration in view of the following comments.

Claim 21 recites a system that includes:

“a storage management stack having a plurality of stack levels, wherein the stack levels include a lowest level identifying one or more storage devices of a first storage environment; plug-in modules wherein each plug-in module interfaces with **a different** one of the stack levels to resolve a reference to a storage object and to pass the resolved reference to a next stack level, unless the resolved reference is an absolute reference to the storage object housed on one or more of the storage devices, **wherein each plug-in module resolves references to storage objects according to a different level of abstraction**; and a controller that selectively calls a number of the plug-in modules until the absolute reference is obtained.” (emphasis added).

Applicant submits at least the above highlighted features are neither taught nor suggested by the cited art. In the present office action (page 5), it is stated “Jackowski discloses extensible service provider for controlling the plug-in modules and the controller that selectively calls a number of the plug-in modules (see Figs. 4-5; Fig. 11; abstract and col. 3, line 40 to col. 4, line 48; col. 9, lines 5-19 and col. 14, lines 1-21, Jackowski). It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Bracha as taught by Jackowski to include the claimed features. The motivation of doing so would have been to increase the efficiency of Jackowski’s system by managing and controlling accessibility to objects on plurality operating systems (col. 3, line 40 to col. 4, line 48, Jackowski).” However, Jackowski merely discloses plug-in modules that interface to an extensible service provider in order to provide various extra network services. The extensible service provider appears at a particular layer in a network stack (Figs. 4-5, Jackowski). Jackowski nowhere discloses that the plug-in modules resolve references to storage objects. Rather, Jackowski teaches a number of examples of plug-in module functions including

“The application-classifier plugin resides on client and/or server machines and each collects network statistics for packets originating from or received by the machine. A policy server gathers information from the machines by polling the application-classifier plugin in each client and

reading the collected statistics. Alternately, each client can periodically send the collected information to the policy server. The policy server can then make policy decisions and prioritize network traffic based on the information collected from the clients' application-classifier plugins." (Jackowski, col. 7, lines 44-54).

As may be seen from the above, the application-classifier plug-in provides information to a policy server. However, Jackowski nowhere discloses nor suggests "plug-in modules wherein each plug-in module interfaces with a different one of the stack levels to resolve a reference to a storage object and to pass the resolved reference to a next stack level, unless the resolved reference is an absolute reference to the storage object housed on one or more of the storage devices, wherein each plug-in module resolves references to storage objects according to a different level of abstraction," as is recited in claim 21. Neither does Bracha disclose or suggest these features.

Accordingly, Applicant submits not all of the features of claim 21 are disclosed or suggested by the cited art, either singly or in combination, and a prima facie case of obviousness is not established. Therefore, claim 21 is patentably distinct from the cited art. As each of dependent claims 22-28 includes at least the features of independent claim 21, each of these claims is believed patentable for at least the above reasons.

Applicant submits the application is in condition for allowance. However, should the examiner believe issues remain that would prevent the present application from proceeding to allowance, the below signed representative requests a telephone call at (512) 853-8866 in order to facilitate a speedy resolution.

CONCLUSION

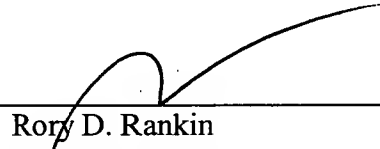
Applicant submits the application is in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5760-17100/RDR.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

Respectfully submitted,



Rory D. Rankin
Reg. No. 47,884
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin,
Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800

Date: July 19, 2006